Styrofoam Projectile Motion Kalman Filter

Noah Harvey

November 24, 2014

Code

main.m

```
# A kalman filter

# initial physical state
# init velocity
v0 = 50;
# initial angle
theta0 = pi/4;
# time interval (1/frequency of camera rate)
t = 1/20;
# counter variable
i = 1;

#-- LOAD DATA -----------------------------------------------------------------
# z =
load can_measure.dat;
# xActual =
load can_actual.dat;
# reduce actual data to something sizeable (I don't need intervals of 0.xx as
# the measured data is every 2.xx)
xActual = xActual(:,1:10:columns(xActual));
#-- END LOAD DATA -------------------------------------------------------------

#-- INIT VARS -----------------------------------------------------------------
# state variable :
#   x,y,dx/dt, dy/dt
X = [0;0;v0*cos(theta0);v0*sin(theta0)];

# state transition model
A = [1,0,t,0;0,1,0,t;0,0,1,0;0,0,0,1];
At = transpose(A);

# control model (assuming gravity is our control variable)
B = [0;(-t^2)/2;0;-t];
u = 9.81;
Bu = B*u;

# init state covariance (4x4 zero matrix)
P = 0*eye(4);

```

```
# prediction covariance
#                     V variance for control signals
Q = 0.001*(B*transpose(B)+eye(4));

# measure covariance
#          V variance for x and y measurements
R = [1,0;0,1]*0.1;

# observation model
H = [1,0,0,0;0,1,0,0];
Ht = transpose(H);

#--- END INIT VARS ------------------------------------------------------------
k_out = [];
# get our current measurement
for zk = z
        #
        #--- Predict ------------------------------------------------------------
        # state
        X = A*X+Bu;
        # covariance
        P = A*P*At+Q;
        #--- END Predict --------------------------------------------------------

        #--- UPDATE -------------------------------------------------------------
        # innovation (measurement) residual
        m_res = zk-H*X;
        # innovation covariance
        Sinv = inv(H*P*Ht+R);
        # kalman gain
        K = P*Ht*Sinv;
        # updated state
        X = X + K*m_res;
        # updated covariance
        P = P - K*H*P;
        #--- END UPDATE ---------------------------------------------------------

        #--- OUTPUT -------------------------------------------------------------
        k_out = [k_out,X(1:2)];
        #--- END OUTPUT ---------------------------------------------------------
end
#--- PLOT ---------------------------------------------------------------------
figure (1);
x = rot90(xActual(1,:),-1);
y = rot90(xActual(2,:),-1);

x0 = rot90(z(1,:),-1);
y0 = rot90(z(2,:),-1);

x1 = rot90(k_out(1,:),-1);
y1 = rot90(k_out(2,:),-1);
```

```
plot(x,y,x0,y0,x1,y1);
#plot(x1,y1);

#-- END PLOT ---------------------------------------------------------------
```